CMPSCI 105: Lecture #12 Searching, Sorting, Joins, and Indexing

PART #1: SEARCHING AND SORTING

Linear Search

- Items can be in any order,
- Have to examine first record, then second record, then third record, etc., until item is found or all items have been examined,
- Worst case search time (item not found) is O(N) for N items,
- Search time grows *linearly* as a function of N.

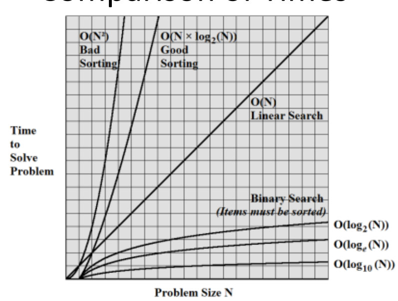
Binary Search

- Items must be sorted on search field,
- Examine middle record, stop if found, but if not found discard half of list known to not contain item, repeat until found or list empty.
- Worst case search time (item not found) is $O(\log_2(N))$ for N items,
- Search time grows *logarithmically* as a function of N.

Sorting

- Very time expensive,
- Worst case sort time is $O(N \times \log_2(N))$ for N items for good sorting algorithms or $O(N^2)$ for bad sorting algorithms,
- Worth it to sort once if binary search can be used many times.

Comparison of Times



"Big-O" Running Times

PART #2: INDEXING vs. SORTING

SORTING

- Physically rearranges table to be in the desired order.
- As we've seen, sorting takes a long time: $O(N \times \log_2(N))$ for N records.

What if we need to see Table in several different orders?

- Option #1: Re-sort table each time a new view is needed.
  - Only one table, but…
  - …takes lots of unnecessary time.
- Option #2: Make several copies of table, each sorted on a different field.
  - Many copies means <u>lots</u> of disk space used, and…
  - Adding/Deleting/Changing record in one means same change must be made to all (data consistency).
- Option #3: Use Indexes.

INDEXING

- Indexes are additional hidden data structures associated with individual fields in a table.
- Indexes are maintained automatically by the database package.
- An index makes a table "look sorted" on the indexed field.
- Every field may have its own index.
- Indexes speed up both searches and joins.

A Table without Indexing

PEOPLE

	ID	NAME
1	12345	Fred
2	72401	Joe
3	22222	Mary
4	54321	Sam
5	20202	Martha
6	11111	Bob
7	47904	Tom

Tables without Indexes

- In a table without any indexes, data records may appear in any random order.
- A table <u>might</u> have its records in a sorted order <u>without</u> an index.
- Searching an unindexed table for matching records requires examining every record in the table. This is a linear search, which runs in O(N) time for N records.

Same Table Indexed on ID Field

PEOPLE

	ID	NAME
1	11111	Bob
2	12345	Fred
3	20202	Martha
4	22222	Mary
5	47904	Tom
6	54321	Sam
7	72401	Joe

Table Indexed on ID Field

- The People table *apparently* is indexed on the ID field.
- People *might be* unindexed, but if it is indexed it would have to be on the ID field.

Searching Indexed Tables

- Typically, indexed fields contain unique information (no duplicates).
- Such fields can be searched with binary search in $O(\log_2(N))$ time. (Check middle item, discard half not containing search term, repeat until found or list becomes empty).

PART #3: JOINS

3A: JOIN TYPES
3B: RELATIONSHIPS

JOINS

- Something spreadsheets can't do.
- Synthesize new tables from two or more source tables.
- Source tables must have a field in common (doesn't have to have the same name, but must have compatible data types).
- Result may contain more records than either source table, or may be empty.

Part 3A: Types of Joins

- Inner Join (intersection of tables)
- Outer Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join (union of tables)

Source Tables

Neither Table has an Index

PEOPLE

	ID	NAME
1	12345	Fred
2	72401	Joe
3	22222	Mary
4	54321	Sam
5	20202	Martha
6	11111	Bob
7	47904	Tom

PAYMENTS

	ID	SALARY
1	60233	$30,000
2	54321	$20,000
3	11111	$40,000
4	97330	$50,000
5	72401	$35,000
6	13333	$45,000

Inner Join on ID Field (Intersection)

Inner Join

	ID	NAME	SALARY
1	11111	Bob	$40,000
2	54321	Sam	$20,000
3	72401	Joe	$35,000

Inner Join

- Contains only records where join-field in one table matches join-field in the other table.
- In this case, there are only three matches. Those three results will be filled in with data from both tables.

Left Outer Join on ID Field

Left Outer Join

	ID	NAME	SALARY
1	11111	Bob	$40,000
2	12345	Fred	
3	20202	Martha	
4	22222	Mary	
5	47904	Tom	
6	54321	Sam	$20,000
7	72401	Joe	$35,000

Left Outer Join

- Contains all data from the left table.
- The matches with the right table will have their records completely filled in.
- Answer records from the left with no match in the right will have “holes”.

Right Outer Join on ID Field

Right Outer Join

	ID	NAME	SALARY
1	11111	Bob	$40,000
2	13333		$45,000
3	54321	Sam	$20,000
4	60233		$30,000
5	72401	Joe	$35,000
6	97330		$50,000

Right Outer Join

- Contains all data from the right table.
- The matches with the left table will have their records completely filled in.
- Answer records from the right with no match in the left will have "holes".

Full Outer Join on ID Field (Union)

Full Outer Join

	ID	NAME	SALARY
1	11111	Bob	$40,000
2	12345	Fred	
3	13333		$45,000
4	20202	Martha	
5	22222	Mary	
6	47904	Tom	
7	54321	Sam	$20,000
8	60233		$30,000
9	72401	Joe	$35,000
10	97330		$50,000

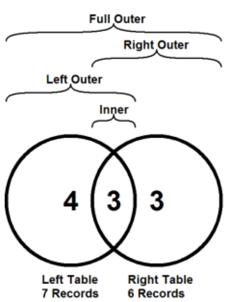
Full Outer Join

- All records from both source tables will be in the answer.
- Those that match will be completely filled in (and appear only once).
- All other records will contain "holes".

For our example

- There are 7 records in the left table.
- There are 6 records in the right table.
- There are 3 matches (the inner join).
- The left outer join will contain 7 records, 3 filled in and 4 with holes.
- The right outer join will contain 6 records, 3 filled in and 3 with holes.
- The full outer join contains 4 (from left) + 3 (inner) + 3 (from right) = 10 records.

Records in Result of Joins



Part 3B: Relationships

- Many:Many
  - Neither table indexed on join field
- 1:Many or Many:1
  - One table indexed on join field
- 1:1
  - Both tables indexed on join field

Many:Many Relationships

- Neither table has an index on the join-field.
- Any record in either table may have multiple matches in the other table.
- <u>Every</u> record in one table is compared against <u>all</u> records in the other table.
- For M records in one table and N records in the other, there will be O(M×N) comparisons.
- It is possible to have M×N records in the answer (all match), or none (no match).

Many:Many Join

PEOPLE

	ID	NAME
1	12345	Fred
2	72401	Joe
3	22222	Mary
4	54321	Sam
5	20202	Martha
6	11111	Bob
7	47904	Tom

PAYMENTS

	ID	SALARY
1	60233	$30,000
2	54321	$20,000
3	11111	$40,000
4	97330	$50,000
5	72401	$35,000
6	13333	$45,000

1:Many or Many:1 Relationships

- One table (M records) is without an index, the other (N records) is indexed on the join-field.
- The indexed table typically has unique information in the indexed field.
- Every record in the unindexed table does a binary search into the indexed table.
- A record in the unindexed table has <u>at most one match</u> in the indexed table.
- A record in the indexed table <u>may have multiple matches</u> in the unindexed table.
- There are O(M×$\log_2$(N)) comparisons.

Many:1 Join, Step #1: 60233

Left Table has an Index

PEOPLE

	ID	NAME
1	11111	Bob
2	12345	Fred
3	20202	Martha
4	22222	Mary
5	47904	Tom
6	54321	Sam
7	72401	Joe

PAYMENTS

	ID	SALARY
1	60233	$30,000
2	54321	$20,000
3	11111	$40,000
4	97330	$50,000
5	72401	$35,000
6	13333	$45,000

#1 #2 #3

Many:1 Join, Step #2: 54321

Left Table has an Index

PEOPLE

	ID	NAME
1	11111	Bob
2	12345	Fred
3	20202	Martha
4	22222	Mary
5	47904	Tom
6	54321	Sam
7	72401	Joe

PAYMENTS

	ID	SALARY
1	60233	$30,000
2	54321	$20,000
3	11111	$40,000
4	97330	$50,000
5	72401	$35,000
6	13333	$45,000

#1 #2

Many:1 Join, Step #6: 13333

Left Table has an Index

PEOPLE

	ID	NAME
1	11111	Bob
2	12345	Fred
3	20202	Martha
4	22222	Mary
5	47904	Tom
6	54321	Sam
7	72401	Joe

PAYMENTS

	ID	SALARY
1	60233	$30,000
2	54321	$20,000
3	11111	$40,000
4	97330	$50,000
5	72401	$35,000
6	13333	$45,000

#3 #2 #1

1:1 Relationships

- Both tables have an index on the join-field.
- Joining the tables requires only one pass each over the two tables.
- A record in either table has <u>at most one match</u> in the other table.
- There are at most O(M+N) comparisons.

1:1 Join

Both Tables have Indexes

PEOPLE

	ID	NAME
1	11111	Bob
2	12345	Fred
3	20202	Martha
4	22222	Mary
5	47904	Tom
6	54321	Sam
7	72401	Joe

PAYMENTS

	ID	SALARY
1	11111	$40,000
2	13333	$45,000
3	54321	$20,000
4	60233	$30,000
5	72401	$35,000
6	97330	$50,000

Suppose M = N = 1024 Records

- Many:Many
  - M×N = 1,048,576 comparisons
  - Possibly 1,048,576 records in answer
- 1:Many or Many:1
  - M×$\log_2$(N) = 1024×10 = 10,240 comparisons
- 1:1
  - M+N = 2048 comparisons (maximum)

Conclusion

- Indexing speeds up both queries and joins.
- In a join, having one index is faster than having none, having two indexes is even faster.
- For many:many relationships, neither table will have an index so joins will be slow (oh, well), and may generate many more records than are in either source table.